Intelligent Agents a New Management Style

Raul Oliveira, Jacques Labetoulle
Corporate Communications Department
Eurecom Institute
06904 SOPHIA ANTIPOLIS CEDEX, France.
email: {oliveira — labetoul }@eurecom.fr

Abstract

In this paper, we present a new management style based on the infiltration of Intelligent Agents into the networked environment. In our perspective, intelligent agents are used to assist users or applications as must as possible. We present solutions to make intelligent agents aware of user requirements, so that the appropriate management solutions could be taken. Without reducing intelligent agents inherent autonomy, we also introduce ways to use domains and policies to influence intelligent agents behavior. We also propose an adequate basis for information and execution models for our management framework.

1 Introduction

The first impression that this paper might create in our minds is that we are just moving tasks from humans beings to softbots (software robot) [1]. We believe that, in the field of network and distributed systems management, the automation process will allow to create new tasks, one could not think of before, and to enhance existing ones.

Even from a Corporate view point there is no intention to replace network management staff. As networks get more complex (e.g. virtual LANs), and the range of offered services broadens, it becomes increasingly difficult to rely solely on operators expertise for proper configuration and operation. This also means that the quality of the management system will no longer depend on the skill level of the person using it as it is the case today.

In section 1, we address some issues that could benefit from the introduction of autonomous intelligent agents. In section 2, we present a new management style built on well scoped domains, ruled by policies, and where autonomous software entities can find the motives to create their own management goals. In section 3, we present the overall management information model that we propose. Finally in section 4, we address the execution environment issue of the management architecture.

User assistance

In the close future we will see an important population of computer users having no insight awareness. They will use complex networks systems and should be provided with good assistance. In contrary to what we knew until now, where most users were able to do some network debugging, and sometimes giving good hints to network managers. For example, a well known message such as "DNS lookup failure", might not provide any meaningful information to an average Netscape user. This message should preferably be sent to autonomous network assistant that knows how to handle it.

[1]a softbot is an agent that interacts with a software environment by issuing commands and interpreting environment's feedback [4]

As networks spread connecting the entire world, building the so called Global Village, some kind of intelligent behavior is expected from these networks. This intelligence will guarantee quality of service to users. The intelligence may be enhanced by improving the capability of the network to assist users by understanding their behavior.

Satisfy user or applications requirements

End users expect network services to be managed in a way that may consistently afford their applications primary requirements. This might seem a difficult goal to achieve, because users or applications do not usually have any entity in the network to which to address their requirements. Hence, creating such a network entity is important. An encouraging sign is the recent protocols like ATM [3] and RSVP [14], which integrate resource allocation and handling of quality of service constraints.

Stand alone applications are the most exigent users that a network can support. Industrial environments are good examples where we find a set of applications using a distributed communication platform and the underlying network almost in stand alone. Our main concern is that those applications are often built either considering the highest QoS [2] (Quality of Service), or alternatively, they are provided with monitoring capabilities, so they are able by themselves to be aware of network and services QoS. The former approach is not pragmatic, and could entail severe damages to the application behavior when the required QoS is not anymore available. The second approach, even if quite usual, does not make sense when the management infrastructure is also monitoring network services and resources.

There is in our opinion a strong requirement to find ways to establish a closer relationship between Network and Systems Management and the end user applications [8], whether these application are distributed or not.

2 New Management Style

So far, we presented the requirements, that do not appear in current management architectures and paradigms. We are proposing a management framework in which a high degree of automation is required, and where a need to adapt the management automatically to the current situation is the critical issue. Probably, these requirements will also lead to management applications designed with a high degree of autonomy, able to decide by themselves what to do. Whether in terms of what to manage, or in terms of the information to exchange with remote management applications.

There is a need for a new management style, built around autonomous and intelligent communicating software entities, able to interact with other less common network actors such as users, applications, service providers, and hosts.

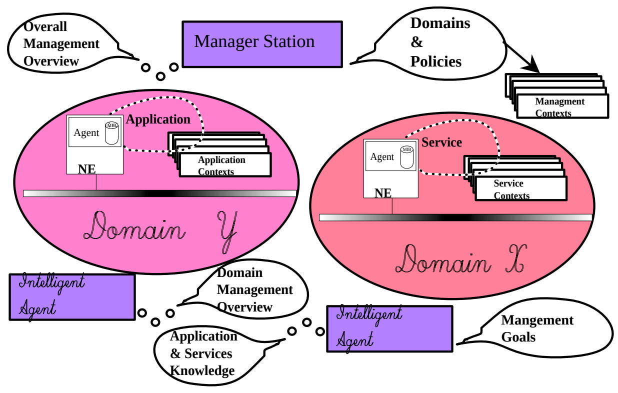

Figure 1:Network actors and management concepts overview.

Figure 1 represents our management style, we have just described above.

[2] ideal computer network not submitted to faults

Intelligent Agents

Several reasons led us to choose Intelligent Agents (IA) as the reference entity around which we built our new management style.

First, Intelligent Agents are not necessarily software components only. They include human beings as well. This means that multi-agents systems are of prime concern to CSCW (Computer Supported Cooperative Work), in which also human agents cooperate and communicate to achieve common goals. This is what we expect to happen in our management framework. Autonomous management applications (Intelligent Agents) should cooperate and communicate with users, applications, network operators, and all a unique goal in mind: enhance the individual Quality of Service (QoS) of applications running over computer networks and distributed systems.

A second major reason is that agent-worthy tasks are generally referred as tasks that require: adaptation, autonomy and asynchrony amongst other attributes. These are the characteristics we had identified so far for our autonomous management applications, in order to satisfy management requirements.

Moreover, some authors refer some basic principles that also instigated our choice. For instance, Wooldrige [13] considers an IA a system that is situated in a dynamic environment, of which it has an incomplete view, and over which it can exert partial control through the performance of actions. Jennings [6] classifies an agent as a self contained problem solving entity which exhibits the following properties: autonomy, social ability, responsiveness and pro-activeness. To Rao [10] IAs will typically be allocated several (possibly conflicting) tasks, and will be required to make decisions about how to achieve these tasks in time, for these decisions to have useful consequences.

It is important to note that IAs will not operate on their own. Rather, they will be always beneath a superior administrative authority, which will provide the IAs with its action scope (management domains) and the policies to which the IAs behavior must obey.

Management Domains and Policies

The use of autonomous management applications, and in our case IAs, requires the use of methodologies to structure networked environments. Domains seem to be the appropriate solution to organize IAs deployment over a distributed environment. Domains provide an excellent means to define boundaries of management responsibility and authority, according to several criteria such as location, physical network connectivity, structuring of the distributed system, organization, function or policy based.

Sloman [12] states that a domain is an object that represents a collection of objects which have been explicitly grouped together to apply a common management policy. Based on this principle one possibility to indicate domain responsibilities to IAs would be to instantiate domain objects and explicitly add references to domain member objects, as proposed for autonomous managers in [2]. However, for what concerns IAs, we think that the right way of specifying a domain should be rule based. Each IA is built with basic rules to constitute domains. Based on these rules, and on others downloaded afterwards, a manager must be able to describe domain instances and the members of each domain.

The subtlety here, in comparison with other domain frameworks, is that it is up to the IA to determine and associate the objects that belong to each domain. We have to remember that networking environments are very dynamic. For this reason we believe not that a manager or another entity could be permanently informing the IA about what are the members of each management domain.

After defining the IA management boundaries it is also necessary to find mechanisms to influence IAs while performing their management activities. The inputs that an agent needs to develop a perception of its surrounding dynamic environment come from applications (users), service providers, other Intelligent Agents or managers. Based on this perception (and on behalf of his superior administrative authority) the Intelligent Agent should be able to create its own activities which we hereafter name IA Management Goals.

The way we choose to express this authority downwards to the IA is based on policies. As

Moffet [7] states, there are policies that motivates activities and policies that give authority to carry out activities. Thus, there is a need to represent and manipulate policies within the management applications such as managers and Intelligent Agents.

To visualize how we plan to influence an IA by using policies, we give the following examples. Imagine an IA whose expertise currently includes managing a DFS (Distributed File System). To this IA we assign an organizational Domain (e.g. CAD Workgroup), and we want the IA to accept the following policy:

- "Verify availability of all file-systems belonging to your administrative domain, from all hosts inside the domain".

We can suppose another scenario in which users are able to express their own requirements to the IAs. Now the IA could be aptless to manage DFSs. Nevertheless, the manager sends it the following policy:

- "Accept users demands to verify availability of file-systems belonging to your administrative domain"

This last approach, first supposes that interactions between users and the management infrastructure are possible. Second that the IA is able to gain aptitude in managing DFSs. Finally this type of approach also leads to an optimization of IA resources, since the IA only needs to check the availability of file-systems currently in use, as specified in application requirements.

Goal creation in motivated agents

When an autonomous agent is required to interact with an environment that is not entirely predictable, a static list of goals is not enough flexible to represent the agent purposes. The state of the environment may change at any time such that pursuing a goal may no longer be realistic, required, or even possible. A single goal may need to be satisfied more than once, or periodically, depending on how the environment, the agent and the relationship between them (i.e. domain) change over time [5].

In the framework proposed in this paper, policies play the role of motivations, that exist to influence an IA so that it focuses their attentions on some types of activities. Application requirements on their turn are the motives that IAs will receive to create and activate goals. In other words an IA will only create a goal, if it has the motivation to do so.

In fact the first type of policy plays both the role of motivation and the role of a motive. After receiving a policy of the first type, the IA can immediately create and activate the associated management goals. While for the second type the IA just stays motivated to create goals on a particular management ontology. That is why we will use the second type of policies presented above, as they seem more appropriate to Intelligent Agents.

A motivated agent is thus typically driven by a number of policies (motivations) that have the capacity to influence the generation of motivated goals in response to detected changes in the domain (application requirements). A motivated agent is thus capable of [5]:

- generating goals on the fly in response to a changing environment,
- altering its focus of attention as its goals and the priorities of those goals change,
- limiting its focus of planning attention such that it is never overloaded by goals demanding resources.

3 Management Information Model

The management framework we presented must be supported by a consistent information model, where all network actors could be integrated. The challenge is to design such an information model that satisfies all actors, given that they are not all concerned with the same environment features.

We choose to rise the abstraction to a level where all the actors, directly or indirectly concerned with the management, could easily share data, based on the different views that they might have from the management environment. The smallest information unit, understandable by all the actors, on the management environment, is the user service [3], regardless of the technology used to build the service or to perform the service management.

The management information model is hence designed in a service oriented way, and structured in three different axes: requirements, policies, domains, and goals. In fact, when participating in the management, all actors have the service as a reference unit of information. That means that applications specify their requirements with service granularity, and service providers specify their capabilities also in a service oriented way. Intelligent Agents create their own management goals in terms of services, and managers will deliver policies and domains also in a service oriented way.

This approach enables a global view where all actors understand each other, and leads the decomposition of the service oriented view to each actor, according to its needs. Moreover, we distinguish management targets from implementation details such as management protocols, structures of management information and the definition of managed objects.

The need for such an information model seems to be a trivial issue, but looking at the several MIBs defined whether under the Internet based standards (SNMP) or even under OSI (CMIP) the reader will easily understand our concern. For instance, the most spread MIBs in the market place, such as the de facto standard MIB-II or Host Resource MIB do not have any user service oriented feature.

Applications and Service Providers Contexts

In [8] we chose contexts to let applications inform Intelligent Agents of their requirements, while in [9] we organized this context in a service oriented way. For each service the application add the QoS dimensions [1] that match the applications requirements. Each of these dimensions could be made up of several QoS domains which finally are composed of sets of attributes, that characterize the QoS required for the service. The number of flavors for QoS dimensions could be high, as well as the number of QoS domains of each dimension. So applications have to publish their QoS requirements as a sequence of needed QoS dimensions and for each QoS dimension a sequence of QoS domains.

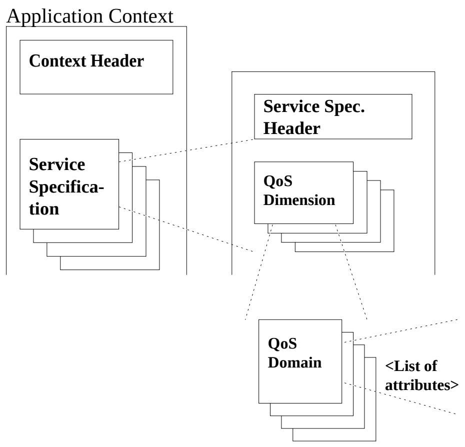

Figure 2:Organization of contexts by service and associated QoS

In most network environments the application will also specify the resources associated with the user service and as well as the identification of the service provider. Nevertheless, this information is optional, since it will be possible in future that some or part of that information will be handled

[3]Hereafter when we say service we will be referring to the user service. Which must not be confused with service primitives of base protocols

by traders, such as anticipated in the ODP framework [11] (which we think is an important reference to build distributed systems).

For what concerns service providers we also preview the use of contexts, in order that the management infrastructure could be informed of who is offering the services, and what is the offered QoS capability. These contexts are structured in the same form as they were for applications. For instance, a service provider can specify the QoS dimensions that might be useful to know the capabilities of the service provider or to help monitoring its behavior.

Publishing is the way we choose to make applications or service providers inform the management front-ends [4] of their requirements or offer capability. This type of interaction was preferred to the traditional discovering process, since discovering entails a waste of bandwidth or processing resources, to obtain the equivalent information.

There was, however, an important drawback in this framework concerning legacy applications and service providers. The solution we design for those entities was to build small satellite processes which will forward the contexts on behalf of application or service providers. For service providers we preview another type of solution. In this case it is the network host on which these entities are installed which takes the responsibility of forwarding these contexts in what we defined as system contexts, since they can carry more than one service provider contexts.
In some cases where a service is composed of sub-services distributed over the network, the approach taken for satellite processes is reused and we have a kind of master process in charge of service context publication.

Applications or service providers can specify in their contexts several services, QoS dimensions and QoS domains. It is possible that the Intelligent Agent facing an application does not have the knowledge to handle all the context content. To avoid situations where actors are not able to understand each other, the first step is a negotiation act between the application and the Intelligent Agent. The knowledge of which information can be handled by the Intelligent Agent, is then used by the application to define and publish its context. From the Intelligent Agent side the negotiation process an opportunity the IA has to obtain this knowledge before the application publishes its context.

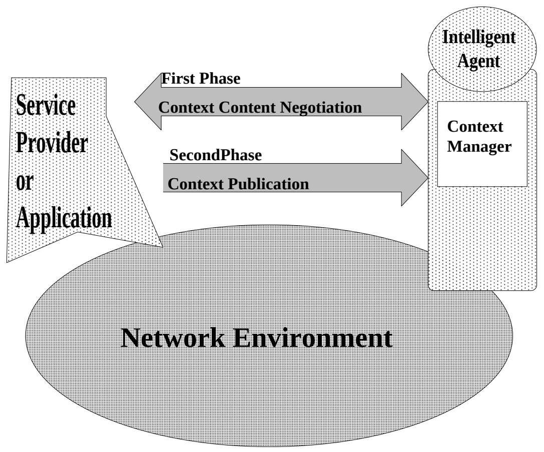

Figure 3:Context content negotiation and publication.

Manager delivered policies

Obligation and authorization are the two most important policies types referred in reference frameworks such as IDSN and SysMan [2]. These policies can be used in a negative or positive sense, for obliging or deterring, and authorizing or prohibiting respectively. We think that these two types of policies are not enough when dealing with Intelligent Agents. Intelligent Agents rather than being obliged or authorized should preferably be motivated, which is more appropriate.

[4] in [8] this was how the Intelligent Agents were named in the presented management infrastructure

Since we have accepted these three policies flavors as sufficient, we explain why and how do we need these policies to deal with intelligent Agents. Obligation policies are useful when we need to ask an IA to perform one or more management actions over determined targets. Motivation policies will be used to create propensity to execute some type of management actions as soon as they are required. The decision of the need to execute these actions is up to the Intelligent Agent based on the perception it has from the networked environment. Unless any reason we do not foresee now, we will not use authorization policies to directly influence Intelligent Agents behavior. In our opinion, as soon as an IA receives obligation or motivation policies it is already and implicitly authorized to perform the associated management operations. However, authorization policies are still of paramount importance to specify which network actors, whether in a management or user role, belonging to local or foreign domains, can request management operations from an Intelligent Agent.

As we made for application or service providers contexts in our framework we try to avoid considering particular features of the different management environments. So again we will specify management policies oriented to service abstraction instead of defining policies applyable directly over real managed objects.

As we stated in [9] management policies and domains specifications will be carried from managers to Intelligent Agents in information structures called Manager Contexts. Figure 4 shows how these manager contexts are organized and structured.

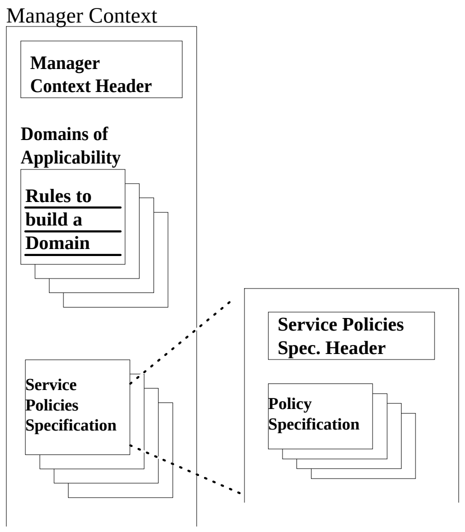

Figure 4:Organization and structure of manager contexts

Intelligent Agents goals

Intelligent Agents goals represent the current Intelligent Agent management intentions. Management Goals are, therefore, a possibility that managers or network operators dispose to monitor IAs current management activities. So they should be available to managers or others IAs on demand.

Intelligent Agents Goals are in other words a kind of high level semantics to describe management operations, independent of service implementation technology and service management paradigms.

A management goal is created to verify, guarantee or observe if user requirements are being satisfied. The management goal is created expressing not only the requirement contexts but also the offering contexts, and it is this aggregation that allows to create management goals that take in account the dynamics of the networked environment.

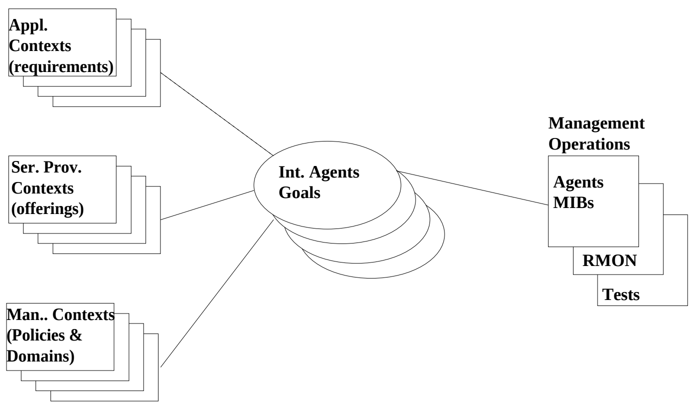

Figure 5:Intelligent Agent goals as relationships between contexts and management operations

Intelligent Agent goals are in fact a relationship between contexts and management operations. They can be characterized according to the time frame for which they are active (ephemeral, continuous, or random), or according to the type of activity (calculate metrics, checking, monitoring, obtain baselines, analyze health and testing). These activities should be built up of standard management operations: read/write MIBs variables, programming RMONs and service testing (by behaving as a common service client).

4 Management Execution Environment

To build the management environment upon which this framework is supported, several design decisions were made, with several constraints in mind: preserve existent management protocols to access base Agents, find a common communication paradigm that supports the information model, and enable management code migration between framework participants.

The first constraint was the easiest to satisfy, since our management framework aims at being independent of the underlying management protocols and paradigms. Being however possible to include and offer mappings from IA management goals to each existing management paradigm [5]. The second constraint, on the contrary, was more controversial, because we were obliged to chose a communication paradigm that might take some time before becoming widely accepted. Nevertheless, the complexity of our framework almost oblige to such a choice. We chose in fact the CORBA communication paradigm, because mappings for the most common programming languages are already available and more will be in the future. This will enable application developers to easily integrate application contexts in their applications, as they need to make the applications participate in the management environment, as previewed in our framework. Finally and in what concerns the core management infrastructure we chose Java as the programming environment. The main reasons to this choice were based on the facilities that this programming environment provides: code migration, portability across multiple platforms, scalability of management infrastructure (by enabling late addition of new Java packages and Java management classes) without needing to rebuild the complete management infrastructure.

The most important communication acts in our framework occur between:

- Applications and IAs
- Service providers and IAs
- IAs themselves
- Managers and IAs

[5]Right now only SNMP have been integrated in the execution environment, because is the management protocol mostly used in enterprise networks to which this framework basically oriented

In order to allow these communications acts to take place, IDL interfaces were specified so that these entities can exchange information to: negotiate context content, publish contexts, exchange management goals, and to deliver management contexts.

Another type of communication is also needed to update IAs capabilities, for what concerns:

- Contexts interpretation
- Goals creation
- Service Models
- Mapping between Goals and Management Goals

These capabilities are in fact object classes, that can be individual object classes or complete object packages. These classes will be transferred, using the FTP protocol, between trusted servers and the IA.

5 Conclusion

Using Intelligent Agents is a must in today research, and for instance, network and distributed systems management is not an exception. Intelligent Agents constitute a research challenge in order to cope with complex environments. Naturally, network and distributed systems management is an ideal application domain.

In this paper we outlined a framework, for which pragmatic solutions were designed, to introduce Intelligent Agents in the management field. We conceive Intelligent Agents to play a network "invisible" assistant role, therefore we developed ways, such as the applications contexts, in order that network or distributed systems actors can publish their requirements or offering capabilities. Although Intelligent Agents are generally viewed as completely autonomous entities, we proposed domains and policies not to reduce their autonomy but to guide or influence their behavior. In what concerns domains, we proposed a rule based domain specification, and for policies we introduced a motivation policy as the policy mostly adapted to Intelligent Agents. An information model oriented to the user service abstraction was introduced so that all network actors could be integrated in the management framework independently of underlying technologies.

A management execution environment, based on the CORBA communication paradigm, is proposed, though we did not adopt any mappings either between IDL and SNMP or IDL and GDMO/CMIP. Instead, we chose to create mappings between IAs management Goals and management operations based on existing management protocols, used to access Agents on network elements.

References

[1] TINA Consortium. Quality of Service Framework, Draft TINA Report, November 1994.

[2] Domain and policy service specification. In K. Becker, U. Raabe, M. Sloman, and K. Twidle, editors, *ISDM Deliverable D6, SysMan Deliverable MA2V2*, October 19 1993.

[3] Martin. de Prycker. ASYNCHRONOUS TRANSFER MODE; SOLUTION FOR BROADBAND ISDN, 1991. CIP (Dec. 91) TK5103.5.P79 1991.

[4] O. Etzioni, N. Lesh, and R. Segal. Building softbors for unix. In Etzioni O, editor, *Software Agents - Papers from the 1994 Spring Symposium*, pages 9–16. AAAI Press, 1994.

[5] Timothy J., Norman, and Derek Long. Alarms: An implementation of motivated agency. In Michael Wooldridge, Jorg P. Muller, , and Milind Tambe, editors, *Intelligent Agents II - Agent Theories, Architectures, and Languages*, pages 219–234, Montreal, August 19-20 1995. Springer.

[6] N. R. Jennings and M. Wooldridge. Applying Agent Technology. *Journal of Applied Artificial Intelligence*, special issue on Intelligent Agents and Multi-Agent Systems, 1995.

[7] Jonathan D. Moffet. *Network and Distributed Systems Management*, chapter 17, Specification of Management Policies and Discretionary Access Control, pages 455–481. Addison-Wesley Publishing Compagny, University of York, jdm@minster.york.ac.uk, 1994.

[8] Raul Oliveira and Jacques Labetoulle. Intelligent agents : a way to reduce the gap between applications and networks. In J. D. Decotignie, editor, *Proceedings of the First IEEE International Workshop on Factory Communications Systems - WFCS'95*, pages 81–90, Leysin, Switzerland, October 4-6 1995.

[9] Raul Oliveira, Dominique Sidou, and Jacques Labetoulle. Customized network management based on applications requirement. In *To appear in Proceedings of the First IEEE International Workshop on Enterprise Networking - ENW '96*, Dallas, Texas, USA, June 27 1996.

[10] A. S. Rao and M. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 321–319, S. Francisco, CA, June 1995.

[11] Basic Reference Model of Open Distributed Processing, ISO DIS 10746-1, ITU-TS Recommandation X.901, April 1994.

[12] Morris Sloman and Kevin Twidle. *Network and Distributed Systems Management*, chapter 16, Domains: A Framework for Structuring Management Policy, pages 433–453. Addison-Wesley Publishing Compagny, Imperial College of Science Technology and Medicine, m.sloman@doc.ic.ac.uk, 1994.

[13] M. Wooldridge. A logic of bdi agents with procedural knowledge. In J. L. Fiadeiro and P.-Y. Schobbens, editors, *Proceedings of the Second Workshop of the MODELAGE Project*, Sesimbra, Portugal, January 15-17 1996.

[14] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, September 1993.